Q1. Write a program to distinguish between Array Indexing and Fancy Indexing.

In [ ]:
```
import numpy as np


arr = np.random.randint(0,9,size=(9),dtype=int)
print(arr)

#select a single element
simple_indexing = arr[6]

print("Simple Indexing: arr[6] -->", simple_indexing)

# select multiple elements
fancy_indexing = arr[[1, 2, 5, 7]]

print("Fancy Indexing: arr[[1, 2, 5, 7]]-->",fancy_indexing)
```

```
[4 4 7 3 5 3 3 7 1]
Simple Indexing: arr[6] --> 3
Fancy Indexing: arr[[1, 2, 5, 7]]--> [4 7 3 7]
```

Q2. Execute the 2D array Slicing.

In [ ]:
```
import numpy as np
arr = np.random.randint(0,9,size=(2,5),dtype=int)

print(arr)
print("output for: arr[0, 1:3]")
print(arr[0, 1:3])
print("output for: arr[1, 1:4]")
print(arr[1, 1:4])
print("output for: arr[0:2, 2:5]")
print(arr[0:2, 2:5])
print("output for: arr[0:1, 2]")
print(arr[0:1, 2])
```

```
[[2 6 4 1 0]
 [4 7 7 1 5]]
output for: arr[0, 1:3]
[6 4]
output for: arr[1, 1:4]
[7 7 1]
output for: arr[0:2, 2:5]
[[4 1 0]
 [7 1 5]]
output for: arr[0:1, 2]
[4]
```

Q3. Create the 5-Dimensional arrays using 'ndmin'.

In [ ]:
```
import numpy as np

arr = np.array([1, 2, 3, 4, 1, 2, 3, 4], ndmin=5)

print(arr)
print('number of dimensions :', arr.ndim)
```

```
[[[[[1 2 3 4 1 2 3 4]]]]]
number of dimensions : 5
```

Q4. Reshape the array from 1-D to 2-D array.

In [ ]:
```
import numpy as np

arr = np.random.randint(0,9,size=(12),dtype=int)
print("input array")
print(arr)
newarr = arr.reshape(4, 3)
print("output array")
print(newarr)
```

```
input array
[4 6 0 6 5 1 8 5 3 7 6 5]
output array
[[4 6 0]
 [6 5 1]
 [8 5 3]
 [7 6 5]]
```

Q5. Perform the Stack functions in Numpy arrays – Stack(), hstack(), vstack(), and dstack().

In [ ]:
```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.stack((arr1, arr2), axis=0)

print("stack() function to concat two 1-d arrays along with axis: \n",arr)

arr = np.hstack((arr1, arr2))

print("hstack() function to concat two 1-d arrays along with x-axis\n",arr)

arr = np.vstack((arr1, arr2))

print("vstack() function to concat two 1-d arrays along with y-axis\n",arr)

arr = np.dstack((arr1, arr2))

print("dstack() to stack along height, which is the same as depth\n",arr)
```

```
stack() function to concat two 1-d arrays along with axis:
 [[1 2 3]
 [4 5 6]]
hstack() function to concat two 1-d arrays along with x-axis
 [1 2 3 4 5 6]
vstack() function to concat two 1-d arrays along with y-axis
 [[1 2 3]
 [4 5 6]]
dstack() to stack along height, which is the same as depth
 [[[1 4]
  [2 5]
  [3 6]]]
```

Q6. Perform the searchsort method in Numpy array.

In [ ]:
```
import numpy as np
```

```
arr1 = np.array([1,2,3,4,5,6,7,5])

x = np.searchsorted(arr1, 7, side='right')

print(x)

y = np.searchsorted(arr1, [7,5])

print(y)
```

```
8
[6 4]
```

Q7. Create Numpy Structured array using your domain features.

In [ ]:
```
import numpy as np

a = np.array([('Elephant', 20, 21.0), ('Python', 14, 29.0), ('Wild Cat', 17, 39.0)],
       dtype=[('name', (np.str_, 10)), ('age', np.int32), ('weight', np.float64)])

print(a)
```

```
[('Elephant', 20, 21.) ('Python', 14, 29.) ('Wild Cat', 17, 39.)]
```

In [ ]:
```
# Sorting according to the name
b = np.sort(a, order='name')
print('Sorting by name', b)

# Sorting according to the age
b = np.sort(a, order='age')
print('\nSorting by age', b)

# Sorting according to the Weight
b = np.sort(a, order='weight')
print('\nSorting by weight', b)
```

```
Sorting by name [('John', 17, 39.) ('Kim', 20, 21.) ('Sam', 14, 29.)]

Sorting by age [('Sam', 14, 29.) ('John', 17, 39.) ('Kim', 20, 21.)]

Sorting by weight [('Kim', 20, 21.) ('Sam', 14, 29.) ('John', 17, 39.)]
```

Q8. Create Data frame using List and Dictionary.

In [ ]:
```
import pandas

mydataset = {
  'Animals': ['Elephant', 'Fox', 'Wolf', 'Wild Cat'],
  'Born': [1972, 1979, 1996, 1991]
}

myvar = pandas.DataFrame(mydataset) # Creating Dataframe from Dictionary

myvar
```

Out[ ]:

	Animals	Born
0	Elephant	1972
1	Fox	1979
2	Wolf	1996
3	Wild Cat	1991

In [ ]:
```
import pandas as pd
list1 = ['Elephant', 'Fox', 'Wolf', 'Wild Cat']

var1 = pd.DataFrame(list1, index =[1,2,3,4], columns =['Animal']) # Dataframe using list with indexing

var1
```

Out[ ]:

	Animal
1	Elephant
2	Fox
3	Wolf
4	Wild Cat

Q9. Create Data frame on your Domain area and perform the following operations to find and eliminate the missing data from the dataset.

- isnull()
- notnull()
- dropna()
- fillna()
- replace()
- interpolate()

In [ ]:
```
import pandas as pd
import numpy as np

df = pd.DataFrame({"animal": ['Lion', 'Tiger', 'Aligator'],
                   "name": [np.nan, 'Deon', 'Bulla'],
                   "born": [pd.NaT, pd.Timestamp("1940-04-25"),
                            pd.NaT]})

# using isnull() function
df.isnull()
```

Out[ ]:

	animal	name	born
0	False	True	True
1	False	False	False
2	False	False	True

In [ ]:
```
# using notnull() function
df.notnull()
```

Out[ ]:

	animal	name	born
0	True	False	False
1	True	True	True
2	True	True	False

In [ ]:
```
# using dropna() function
df.dropna()
```

Out[ ]:

	animal	name	born
1	Tiger	Deon	1940-04-25

In [ ]:
```
# using dropna(axis='columns') function
df.dropna(axis='columns')
```

Out[ ]:

	animal
0	Lion
1	Tiger
2	Aligator

In [ ]:
```
# using fillna() function
df.fillna(0)
```

Out[ ]:

	animal	name	born
0	Lion	0	0
1	Tiger	Deon	1940-04-25 00:00:00
2	Aligator	Bulla	0

In [ ]:
```
# using replace() function
df.replace(np.NAN, " ")
```

Out[ ]:

	animal	name	born
0	Lion		
1	Tiger	Deon	1940-04-25 00:00:00
2	Aligator	Bulla	

In [ ]:
```
#using interpolate()
df.interpolate(method='bfill')
```

Out[ ]:

	animal	name	born
0	Lion	Deon	1940-04-25
1	Tiger	Deon	1940-04-25
2	Aligator	Bulla	NaT

Q10. Perform the Hierarchical Indexing in the above created dataset.

In [ ]:
```
df = pd.DataFrame({"id": [1,2,3],
    "animal": ['Lion', 'Tiger', 'Aligator'],
   "name": [np.nan, 'Deon', 'Bulla'],
    "born": [pd.NaT, pd.Timestamp("1940-04-25"), pd.NaT]})

df = df.set_index(['id'])
df.sort_index()
print(df)
df.loc[[2]]
```

```
Hierarchical Indexing
      animal   name       born
id
1       Lion    NaN        NaT
2      Tiger   Deon 1940-04-25
3   Aligator  Bulla        NaT
```

Out[ ]:

	animal	name	born
id			
2	Tiger	Deon	1940-04-25